

APPARATUS AND METHOD FOR DYNAMICALLY DISABLING FAULTY EMBEDDED MEMORY IN A GRAPHIC PROCESSING SYSTEM

TECHNICAL FIELD

The present invention is related generally to the field of computer graphics, and more particularly, to a memory system for use in a computer graphics processing system.

BACKGROUND OF THE INVENTION

Graphics processing systems are included in computer systems to assist a host microprocessor with some of the processing burden involved with rendering graphic images. Graphics processing systems include semiconductor devices having the entire system fabricated on a single semiconductor substrate, as well as peripheral component cards inserted into a computer system and having multiple devices located on a single component card. As computer applications have become more computer graphics intensive, the need for graphics processing systems having the ability to process graphics data and render graphics images at high speeds has also increased. For example, high resolution three-dimensional (3D) computer animation, which at one time was limited to computer workstations, has become standard in the computer gaming industry for home personal computers. Consequently, current graphics processing systems must be capable of processing the 3D graphics data and rendering frames of graphics images fast enough to avoid image stuttering while maintaining image quality.

To this end, graphics processing systems are typically designed with high-speed graphics processors and a limited amount memory fabricated on the same semiconductor substrate as the graphics processor. Such memory has low-access times and is commonly referred to as embedded memory. The use of embedded memory in a graphics processing system has the advantage of providing a limited amount of high-speed memory to facilitate fast graphics processing. The low access times of the embedded memory are in part the result of the physical proximity of the embedded

memory to the graphics processor, as well as the fact that the embedded memory is dedicated for the purpose of graphics processing.

As semiconductor fabrication techniques have improved, and memory cell design has become more sophisticated, the amount of embedded memory available 5 in a graphics processing system has dramatically increased. It is now possible to find a graphics processing system that includes as much as 12 MBytes of embedded memory. However, a consequence of increasing the available embedded memory is that there is now an increased chance for the embedded memory to have defective memory locations. As a result of the embedded memory being located on the same 10 semiconductor substrate as the graphics processor, a defective memory location in the embedded memory would render an otherwise functional graphics processing processor unusable.

Consequently, as with most memory arrays, the embedded memory of a graphics processing system is designed to include redundant memory. The redundant 15 memory is substituted for the defective embedded memory, and thus, increases the production yield of fully functional graphics processing systems. The redundant memory is substituted for the defective memory by re-mapping memory addresses of the faulty memory to the functional redundant memory. The process of testing for faulty memory, and replacing it with redundant memory occurs as part of the 20 manufacturing process of the graphics processing system. The re-mapping of memory addresses occurs internally to the graphics processing system and appears transparent to the computer system.

Although redundant memory has been effective in increasing the production yield of graphics processing devices, as the amount of embedded memory 25 has increased, so has the amount of redundant memory in order to maintain the relative recovery rate from substituting faulty embedded memory with available redundant memory. As a result, more area on the semiconductor substrate is occupied by redundant memory. Where reducing the size of the graphics processing device is desired, allocating space on the semiconductor substrate to redundant memory, that is,

memory that may or may not be used depending on the amount of faulty embedded memory, is not likely to be a preferable alternative.

Therefore, there is a need for a graphics processing system and method where an otherwise functional graphics processing system having more faulty memory
5 than can be substituted by available redundant memory can still be used for graphics processing.

SUMMARY OF THE INVENTION

The invention is directed to a memory subsystem and method for ignoring faulty memory sub-arrays and assigning functional memory sub-arrays to
10 accessible memory blocks. The memory subsystem includes a memory array segmented into a plurality of memory sub-arrays and a memory controller that accesses the memory array. The memory controller references a register storing a pointer value that assigns a memory block to each of the functional memory sub-arrays of the memory array. The faulty memory sub-arrays are left unassigned so that they are ignored by the memory
15 controller. Upon receiving a memory access request to a particular memory block, the memory controller accesses the functional memory sub-array assigned to the particular memory block by the pointer value.

The memory subsystem may further include a second memory array and a second memory controller coupled to the first memory controller through a memory
20 controller bus. The functional memory sub-arrays of the second memory array are also assigned to memory blocks. A respective start address and size value define the addressable memory area of each of the memory arrays. Each memory controller references the respective start address and size value upon receiving a memory access request to determine whether the particular memory block is assigned to a sub-array
25 within the memory array to which it is coupled. The memory access request is passed to the other memory controller over the memory controller bus if it is determined that the memory block is not within the addressable memory area.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a computer system in which embodiments of the present invention are implemented.

Figure 2 is a block diagram of a graphics processing system in the 5 computer system of Figure 1.

Figure 3 is a block diagram representing a memory system according to an embodiment of the present invention.

Figure 4 is a block diagram of a memory system having a distributed memory controller arrangement.

10 Figure 5 is a block diagram representing the memory system of Figure 4 according to another embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Embodiments of the present invention provide a memory system that allows an otherwise functional graphics processing system that has faulty blocks of 15 memory to be used for graphics processing. The faulty blocks of memory are ignored during memory access by programming and storing appropriate values in a register, or registers, that are consulted by a memory controller during memory access.

Certain details are set forth below to provide a sufficient understanding of the invention. However, it will be clear to one skilled in the art that the invention 20 may be practiced without these particular details. In other instances, well-known circuits, control signals, timing protocols, and software operations have not been shown in detail in order to avoid unnecessarily obscuring the invention.

Figure 1 illustrates a computer system 100 in which embodiments of the present invention are implemented. The computer system 100 includes a processor 104 coupled to a host memory 108 through a memory/bus interface 112. The memory/bus interface 112 is coupled to an expansion bus 116, such as an industry standard architecture (ISA) bus or a peripheral component interconnect (PCI) bus. The computer system 100 also includes one or more input devices 120, such as a keypad or a mouse, coupled to the processor 104 through the expansion bus 116 and the memory/bus

interface 112. The input devices 120 allow an operator or an electronic device to input data to the computer system 100. One or more output devices 120 are coupled to the processor 104 to provide output data generated by the processor 104. The output devices 124 are coupled to the processor 104 through the expansion bus 116 and memory/bus interface 112. Examples of output devices 124 include printers and a sound card driving audio speakers. One or more data storage devices 128 are coupled to the processor 104 through the memory/bus interface 112 and the expansion bus 116 to store data in, or retrieve data from, storage media (not shown). Examples of storage devices 128 and storage media include fixed disk drives, floppy disk drives, tape cassettes and compact-disc read-only memory drives.

The computer system 100 further includes a graphics processing system 132 coupled to the processor 104 through the expansion bus 116 and memory/bus interface 112. Optionally, the graphics processing system 132 may be coupled to the processor 104 and the host memory 108 through other types of architectures. For example, the graphics processing system 132 may be coupled through the memory/bus interface 112 and a high speed bus 136, such as an accelerated graphics port (AGP), to provide the graphics processing system 132 with direct memory access (DMA) to the host memory 108. That is, the high speed bus 136 and memory bus interface 112 allow the graphics processing system 132 to read and write host memory 108 without the intervention of the processor 104. Thus, data may be transferred to, and from, the host memory 108 at transfer rates much greater than over the expansion bus 116. A display 140 is coupled to the graphics processing system 132 to display graphics images. The display 140 may be any type of display, such as a cathode ray tube (CRT), a field emission display (FED), a liquid crystal display (LCD), or the like, which are commonly used for desktop computers, portable computers, and workstation or server applications.

Figure 2 illustrates circuitry included within the graphics processing system 132 for performing various three-dimensional (3D) graphics functions. As shown in Figure 2, a bus interface 200 couples the graphics processing system 132 to the expansion bus 116. In the case where the graphics processing system 132 is coupled to the processor 104 and the host memory 108 through the high speed data bus 136 and

the memory/bus interface 112, the bus interface 200 will include a DMA controller (not shown) to coordinate transfer of data to and from the host memory 108 and the processor 104. A graphics processor 204 is coupled to the bus interface 200 and is designed to perform various graphics and video processing functions, such as, but not limited to, generating vertex data and performing vertex transformations for polygon graphics primitives that are used to model 3D objects. The graphics processor 204 is coupled to a triangle engine 208 that includes circuitry for performing various graphics functions, such as clipping, attribute transformations, rendering of graphics primitives, and generating texture coordinates for a texture map. A pixel engine 212 is coupled to receive the graphics data generated by the triangle engine 208. The pixel engine 212 contains circuitry for performing various graphics functions, such as, but not limited to, texture application or mapping, bilinear filtering, fog, blending, and color space conversion.

A memory controller 216 coupled to the pixel engine 212 and the graphics processor 204 handles memory requests to and from an embedded memory 220. The embedded memory 220 stores graphics data, such as source pixel color values and destination pixel color values. The memory controller 216 includes various registers that, as will be explained in more detail below, store values that are used to disable faulty blocks of memory during the access of the embedded memory 220. In this way, a graphics processing system that is otherwise functional, but has a faulty block of embedded memory that cannot be entirely replaced by available redundant memory, can nevertheless still be used for graphics processing.

A display controller 224 coupled to the embedded memory 220 and to a first-in first-out (FIFO) buffer 228 controls the transfer of destination color values to the FIFO 228. Destination color values stored in the FIFO 336 are provided to a display driver 232 that includes circuitry to provide digital color signals, or convert digital color signals to red, green, and blue analog color signals, to drive the display 140 (Figure 1).

Illustrated in Figure 3 is a representation 300 of the memory controller 216 and the embedded memory 220 (Figure 2). The memory controller 216 includes register 304 having fields 312, 316, 320, 324, and 330, and the embedded memory 220

includes an embedded memory array 308 segmented into four memory blocks 334, 338, 342, and 346. The register 304 is described above as being included in the memory controller 216, however, the register 304 is not limited to this location, and may be located externally to the memory controller. The register 304 includes a memory valid field 312 for storing a value indicative of which memory blocks 334, 338, 342, and 346, are functional, and further includes four BANK fields, 316, 320, 324, and 330. Each field corresponds to one of the four blocks of memory 334, 338, 342, and 346. Although the embedded memory array 308 has been illustrated in Figure 3 as being segmented into four memory blocks, it will be appreciated that the embedded memory array 308 may be segmented into greater or fewer memory blocks. Moreover, the particular size of each memory block into which the embedded memory 308 is segmented is a detail that may be changed without departing from the scope of the present invention. As such, the number of memory blocks into which the embedded memory array 308 is segmented, or the size of each of the memory blocks, should not limit the scope of the present invention.

Figure 3a illustrates a situation where it has been determined from functional testing that all four blocks of memory 334, 338, 342, and 346 are functional. The value 1111 is written to and stored in the memory valid field 312 to indicate that all four of the blocks of memory may be used. Next, the values indicating which blocks of memory should be accessed when a memory access request is made to a respective memory bank is programmed and stored in the BANK fields 316, 320, 324, 330. The value 00 is stored for the BANK0 field, indicating that when access is requested to a memory location having a memory address in BANK0, memory block 334 will be accessed by the memory controller 216 (Figure 2). The value 01 is stored for the BANK1 field, indicating that when access to a memory location having a memory address in BANK1 is requested, the memory block 338 will be accessed. The values 10 and 11 are programmed and stored for the fields 324 and 330, respectively, so that the appropriate memory block will be accessed as well.

Figure 3b illustrates a case where the memory block 338 has been determined during function testing to be faulty. The values in the fields of the register

- 304 are programmed so that the memory block 338 is ignored during memory access. That is, the value 1011 is written and stored in the memory valid field 312 to indicate that the second memory block, that is, the memory block 338, is not valid. The values 00, 10, and 11 are programmed for the BANK fields 316, 320, and 324, respectively.
- 5 The value stored in the BANK3 field is 330 does not matter since the values in the memory valid field 312, and the BANK fields 316, 320, and 324 will prevent access to the fourth bank of memory, namely BANK3. The value 00 in the BANK1 field 316 indicates to the memory controller 216 that access to a memory location having an address in BANK0 will be made to the memory block 334. The value 10 in the BANK2
- 10 field 320 indicates that access to a memory location having a memory address in BANK1 will be made to the memory block 342. The value 11 in the BANK2 field 324 indicates that access to a memory location having a memory address in BANK2 will be made to the memory block 346. The memory block 338, which is faulty in the present example, is consequently ignored.
- 15 The present example illustrates that although an entire block of memory may be faulty, and cannot be repaired through the use of redundant memory, the graphics processing system 132 with reduced overall available embedded memory may nevertheless still be used for graphics processing since the faulty block of memory is ignored by the memory controller 216 during memory access.
- 20 Figure 3c illustrates the case where memory blocks 342 and 346 have been determined during function testing to be faulty. The value 1100 is programmed and stored in the memory valid field 312 to indicate to the memory controller 216 that only memory blocks 334 and 338 are valid. The values 00 and 01 are written to the BANK fields 316 and 320, respectively, so that memory access requests to BANK0 and
- 25 BANK1 are directed to the correct memory blocks, namely, memory blocks 334 and 338. The value stored in the BANK fields 324 and 330 are immaterial since the memory blocks 342 and 346 are ignored by the memory controller 216. Although the overall available memory of the graphics processing system 132 as represented in Figure 3c is reduced by two blocks of memory, the graphics processing system 132 may
- 30 still be used for graphics processing.

Ins a17

Figure 4 illustrates a distributed memory controller memory subsystem 400 that may be substituted into a graphics processing system. A more detailed description of a similar memory subsystem is provided in Patent Application No. _____, filed _____, which is incorporated herein by reference. To summarize, the memory subsystem 400 includes two memory controllers 402 and 422 coupled together through a memory controller bus 216. The memory controller bus 216 allows memory access requests, as well as data, to be passed between the two memory controllers 402 and 422. Each of the memory controllers 402 and 422 is coupled to an addressable memory area 412 and 432, respectively, that is defined by two values. The two values are stored in registers 404a-b and 406a-b. Registers 404a and 406a of the memory controller 402 store the start address and memory size for the addressable memory area 412, and registers 404b and 406b of the memory controller 422 store the start address and memory size value for the addressable memory area 432. These values are referenced by the respective memory controller to determine whether a memory access request is to a memory location in the addressable memory area to which the memory controller is coupled.

For example, the arrangement of the memory subsystem 400 allows a memory access request made to the memory controller 402 over request lines 408 to be passed to the memory controller 422 when the requested memory location has a memory address located in the addressable memory area 432. As mentioned previously, the memory controller receiving the memory access request can determine whether the requested address is located within the addressable memory area to which it is coupled by checking the values of the start address and memory size. In the present example, the memory controller 422 receives the memory access request from the memory controller 402, and accesses the addressable memory area 432 to service the memory access request. If the memory access request received by the memory controller 402 is a read command, the memory controller 422 reads the data from the requested memory location and provides the data back to the memory controller that originally received the memory access request, namely, the memory controller 402. If the memory access request was a write command, data to be written to the memory location accompanies

the memory access request that is passed from the memory controller that originally received the memory access request.

JES INS B15

Figure 5 illustrates a memory subsystem 500 according to another embodiment of the present invention. The memory subsystem 500 has a distributed memory controller arrangement similar to that illustrated in Figure 4 represented by registers 504 and 554. Each of the registers 504 and 554 are located in a respective memory controller. The memory subsystem 580 further includes addressable memory areas represented by memory arrays 508 and 558. The memory array 508 is segmented into m memory blocks and the memory array 558 is segmented into n memory blocks.

- 10 The register 504 includes start address field 512, memory size field 514, and memory valid field 516, and a number of BANK fields 518-526. Each of the BANK fields 518-526 corresponds to a memory block in the memory array 508. The register 554 includes start address field 562, memory size field 564, and memory valid field 566, and BANK registers 568-574. Each of the BANK fields 568-574 corresponds to a memory block in
- 15 the memory array 558. The start address field 512 and memory size field 514 are programmed with the start address and memory size for the memory array 508. As explained previously, these values are referenced by the respective memory controllers in order to determine whether to pass a memory access request it receives to another memory controller in order to access the requested memory location.

- 20 The memory valid fields 516 and 566 store a value indicative of the functional memory blocks of the respective memory array. The BANK fields 518-526 and 568-574 store values indicating which blocks of memory will be accessed when a memory access request is made to a respective memory bank. As explained with respect to Figure 3, faulty memory blocks of the memory arrays 508 and 558 may be
- 25 ignored by the memory controllers 504 and 554 by programming the appropriate values in the memory valid fields 516 and 566, and the BANK fields 518-526 and 568-574.

- Although not included in the representation 300 illustrated in Figure 3, the start address and memory size fields of the memory controller 504 and 554 (Figure 5) should be programmed with the appropriate values so that faulty memory blocks may be ignored when the memory arrays 508 and 558 are accessed. For

example, if the memory addresses of the memory arrays 508 and 558 are to appear contiguous, the start address programmed into field 562 should be the next consecutive memory address following the last addressable memory location of the memory array 508. The last addressable memory location can be determined by subtracting one from
5 the sum of the value stored in the start address field 512 and the value stored in the memory size field 514. Where all of the memory blocks of the memory array 508 are functional, the value stored in the memory size field should be equal to the quantity ($m \times$ size of a memory block). Consequently, for the memory addresses of the memory arrays 508 and 558 to appear as a contiguous memory space, the value programmed into
10 the start address field 562 should be [start address + ($m \times$ size of a memory block)]. However, if two memory blocks of the memory array 508 are faulty, then the value in stored in the memory size field 514 should be adjusted accordingly, that is, the memory size value should be programmed to equal [$(m-2) \times$ size of a memory block]. For the memory of memory arrays 508 and 558 to appear contiguous, the value programmed
15 into the start address field 562 should be [start address + ($(m-2) \times$ size of a memory block)]. The values stored in the memory valid field 516 and the BANK fields 518 should also reflect the two faulty memory blocks in order for the faulty memory blocks to be ignored when the memory array 508 is accessed.

The previous example illustrates how the values programmed into the
20 registers 504 and 554 of the memory controllers 402 and 422 (Figure 4) allow an otherwise functional graphics processing system having faulty blocks of embedded memory to nevertheless process graphics data. Moreover, the memory subsystem 500 provides flexibility as to how functional memory may be allocated. The functional memory blocks of the memory arrays 508 and 558 may be allocated in a contiguous fashion, as described in the previous example, or in another desired manner based on
25 the graphics application executing on the computer system 100 (Figure 1).

From the foregoing it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the
30 invention. Accordingly, the invention is not limited except as by the appended claims.